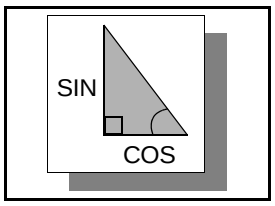
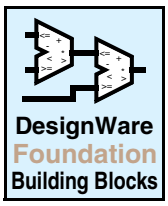




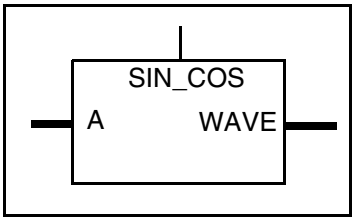
DW_sincos

Sine and Cosine

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

- Parameterized word length
- sine or cosine output by controlling `SIN_COS` port.
- DesignWare Datapath generator is employed for better timing and area.



Revision History

Description

DW_sincos is a fixed point sine and cosine unit that calculates `WAVE` = sin(π`A`) or cos(π`A`) where `A` and `WAVE` are fixed point value by controlling the `SIN_COS` pin. DW_sincos has two additional parameters: *arch* is for the implementation between the area-optimized and the speed-optimized, and *err_range* allows users to choose the error range between 1 ulp error and 2 ulp error.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*A_width* bits	Input	Input data (radian)
SIN_COS	1 bit	Input	Function select: ■ 0: Sine function ■ 1: Cosine function
WAVE	*WAVE_width* bits	Output	Sine or cosine value

Table 1-2 Parameter Description

Parameter	Values	Description
A_width	3 to 34 bits	Word length of `A`
WAVE_width	2 to 35 bits	Word length of `WAVE`
arch	0 or 3 Default: 0	Select the implementation ■ 0: Area-optimized implementation (for behavior that is backward-compatible with DC releases up to R-2020.09) ■ 1: Speed-optimized implementation (for behavior that is backward-compatible with DC releases up to R-2020.09) ■ 2: Area-optimized implementation with *err_range* control ■ 3: Speed-optimized implementation with *err_range* control

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
err_range	1 or 2 Default: 1	Select the error range For *arch* = 0 or 1: ■ The error bound is always 2 ulp, regardless of the *err_range* value (this behavior is preserved to ensure backward-compatibility with DC releases up to R-2020.09) For *arch* = 2 or 3, the following values select the error range: ■ 1: |true value - calculated| < 1 ulp[ab] ■ 2: |true value - calculated| < 2 ulp[ab]

a. 1 ulp corresponds to the weight of the LSB of the output value, which is equivalent to $2^{(-WAVE\_width+2)}$.
b. When *arch* = 2 or 3 and the precision is 25 bits or more, the error can reach 3 ulp.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW02.DW_SINCOS_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_sincos_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_sincos.v	Verilog simulation model source code

Error Behavior

The parameter *err_range* controls the error level at the WAVE output. When *err_range* = 1, the WAVE value has an error of at most $2^{(-WAVE\_width+2)}$, which is the weight of the WAVE LSB representation. For example, with *WAVE_width* = 6 and the long-precision calculation of the function is 01.100011, the output can be 01.1000 or 01.1001. The error is $2^{(-WAVE\_width+2)} = 2^{-4}$.

When *err_range* = 2, the value of WAVE may have a larger error up to $2^{(-WAVE\_width+3)}$. By increasing the error bound, you can reduce the hardware complexity, save area, and reduce the critical path.

Note As explained in Table 1-2, the *err_range* parameter has no effect on the hardware complexity or numerical behavior when *arch* = 0 or *arch* = 1. In these configurations, the error range varies depending on the configuration.

Functional Description

The input angle A is treated as a binary fixed point number which is converted to radians when multiplied by π. When A is interpreted as unsigned, the input angle A is a binary subdivision of the range $0 \le A < 2$. When A is interpreted as signed (two's complement), the range is $-1 \le A < 1$.

Table 1-5 Operations of DW_sincos

SIN_COS input	A input	WAVE output
0	A (unsigned or two's complement)	sin(πa)
1	A (unsigned or two's complement)	cos(πa)

Input Data Format

The input value a is interpreted with the MSB as the only bit to the left of the decimal point. For example, if *A_width* = 6, the input value 111000 is interpreted as 1.11000.

The sine (or cosine) value calculated by DW_sincos is the same whether A is interpreted is an unsigned number or as a signed (two's complement) number. For example, consider the case where SIN_COS = 0 and the input value A is 1.10000. Interpreted as an unsigned number, the decimal equivalent of A is 3/2, the angle is 3π/2, and the sine value is -1 (decimal).

When 1.10000 is interpreted as a signed number, the MSB becomes the sign bit (1 in this example, indicating a negative number) and .10000 is the two's complement of .10000. The decimal equivalent is (-1/2). DW_sincos calculates the sine of (-π/2), which is also -1 (decimal).

Similarly, when SIN_COS = 1, DW_sincos calculates the cosine. If the input value (A = 1.10000) is interpreted as an unsigned number, DW_sincos calculates the cosine of 3π/2, which equals 0. If the input value (A = 1.10000) is interpreted as a two's complement number, DW_sincos calculates the cosine of (-π/2), which also equals 0.

Output Data Format

The output value WAVE is of the form SX.XXXX, where S is the sign bit and X.XXXX is the two's complement of the binary value of the sine or cosine. For example, the sine value 1 (decimal) is represented as 01.0000 (when *WAVE_width* = 6), and the sine value -1 is represented as 11.0000.

Table 1-6 shows the bit assignments for the input and output pins. Table 7 shows input and output values for an example implementation where *A_width* = 6, *WAVE_width* = 6.

Table 1-6 Input and Output Data Format

Pin	Range (binary)	Range (decimal)
A (unsigned)	1.111…111 (< 2) 1.111…110 … 1.000…000 (= 1) … 0.000…001 0.000…000 (= 0)	$0 \leq A < 2$
A (two's complement)	0.111…111 (< 1) 0.111…110 … 0.000…000 (= 0) 1.111…111 (< 0) … 1.000…001 1.000…000 (= -1)	$-1 \leq A < 1$
WAVE (two's complement)	01.000…000 (= 1) 00.111…111 … 00.000…000 (= 0) 11.111…111 (< 0) … 11.000…001 11.000…000 (= -1)	$-1 \leq WAVE \leq 1$

Table 1-7 Sample Data Value for *A_width* = 6 and *WAVE_width* = 6

Value of A	SIN_COS	Function	Value of WAVE
1.00000 (unsigned)	0	sin(π)	00.0000
1.10000 (unsigned)	0	sin(3π/2)	11.0000
1.00000 (two's complement)	1	cos(-π)	11.0000
0.10000 (two's complement)	0	sin(π/2)	01.0000

Alternative Implementation of Sine and Cosine with DW_lp_multifunc

The sine and cosine operation can also be implemented by DW_lp_multifunc component, which evaluates the value of sine and cosine with an error bounded by the weight of the LSB of the output `WAVE`. The value from DW_lp_multifunc and the value from DW_sincos may differ by at most 1 LSB. Performance and area of the synthesis results are different between the DW_sincos and sine and cosine implementation of the DW_lp_multifunc, depending on synthesis constraints, library cells and synthesis environments. By comparing performance and area between the sine and cosine implementation of DW_lp_multifunc and DW_sincos component, the DW_lp_multifunc provides more choices for the best synthesis results. Below is an example of the Verilog description for the sine and cosine of the DW_lp_multifunc. For more detailed information, see the DW_lp_multifunc datasheet.

```
DW_lp_multifunc #(op_width, 24) U1 (
    .a(a),
    .func(func),
    .z(z),
    .status(status)
);
```

Related Topics

- Datapath – Floating Point Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_sincos_inst is
      generic (
        inst_A_width : INTEGER := 24;
        inst_WAVE_width : INTEGER := 25;
        inst_arch : INTEGER := 0;
        inst_err_range : INTEGER := 1
        );
      port (
        inst_A : in std_logic_vector(inst_A_width-1 downto 0);
        inst_SIN_COS : in std_logic;
        WAVE_inst : out std_logic_vector(inst_WAVE_width-1 downto 0)
        );
    end DW_sincos_inst;


architecture inst of DW_sincos_inst is

begin

    -- Instance of DW_sincos
    U1 : DW_sincos
    generic map (
          A_width => inst_A_width,
          WAVE_width => inst_WAVE_width,
          arch => inst_arch,
          err_range => inst_err_range
          )
    port map (
          A => inst_A,
          SIN_COS => inst_SIN_COS,
          WAVE => WAVE_inst
          );


end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_sincos_inst( inst_A, inst_SIN_COS, WAVE_inst );

parameter inst_A_width = 24;
parameter inst_WAVE_width = 25;
parameter inst_arch = 0;
parameter inst_err_range = 1;


input [inst_A_width-1 : 0] inst_A;
input inst_SIN_COS;
output [inst_WAVE_width-1 : 0] WAVE_inst;

    // Instance of DW_sincos
    DW_sincos #(inst_A_width, inst_WAVE_width, inst_arch, inst_err_range) U1 (
                .A(inst_A),
                .SIN_COS(inst_SIN_COS),
                .WAVE(WAVE_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
October 2020	DWBB_202009.1	■ For STAR 9001562033: - In Table 1-2 on page 1: - Updated the lower range of A_width - Added new architecture values for *arch* - Updated the description of *err_range* - Added "Error Behavior" on page 2 - Clarified details in "Alternative Implementation of Sine and Cosine with DW_lp_multifunc" on page 5
January 2020	DWBB_201912.1	■ Corrected port names for DW_lp_multifunc in "Alternative Implementation of Sine and Cosine with DW_lp_multifunc" on page 5
July 2019	DWBB_201903.3	■ Removed reference to minPower library in "Alternative Implementation of Sine and Cosine with DW_lp_multifunc" on page 5
March 2019	DWBB_201903.0	■ Removed minPower designation from this datasheet ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com